Ontologies

Report

DBpedia

<Tuesday, 27 de July de 2010>

- Bruno Tavares; 31528

Index

Abstract

This work aims to gather geographic data from two different databases, Geonames and DBpedia, and calculate various similarities between the different elements, allowing an improvement of these databases and identify the overlap of these databases, allowing complement the information in each database.

Introduction

Ontology

Ontology is a schema to represent the knowledge based on a set of concepts, in a given field, and relations between them. There are several types of ontologies, such as disease (Unified Medical Language System), genes and gene products (Gene Ontology) and geographical (GeoNames) among others.

Ontologies usually share a similar structure; describing elements (instances) based on attributes (features, properties, etc.) and classes (concepts) and stipulate relationships between the various elements and classes. In a geographical ontology, one element could be Lisbon, which has as attributes a given population and geographic location (latitude and longitude), and belongs to a class such as “capital of a country”, and has relations like is-a “city”, which could be part-of “District of Lisbon”/”District of a Country” and in turn also be part-of “Portugal”/”Country”.

DBpedia

DBpedia is a project that aims to gather structured information from Wikipedia, providing access to this information in a Resource Description Framework (RDF) format, using a SQL-likequery language (RDF query language) system called SPARQL (**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage). This database contains more than 3million entries, with almost half of these are in a consistent ontology, such as +300k people, +400k places or 140k organizations. For Portugal, this information is not very structured.

Geonames

GeoNames is a geographic ontology with over 8000000 entries, being categorized into nine classes (A [country, state, region,etc], P [city, village,etc], etc), and nearly 650 subclasses (independent political entity, populated place, etc). Like other ontologies, Geonames has attributes on each element (District of Lisbon), such as its population and geographical coordinates. And the relationships between different elements, in the case of Portugal these relationships are poorly annotated.

Selected Ontologies

The selected ontologies to perform this work were Geonames (reliable source of geographic information) and DBpedia (source constantly updated).

Methodology

DBpedia

Because of the weak structure of the data from DBpedia, the process of collecting such data was implemented in two phases. In the first phase was created the data structure based on the database dump from the Portuguese version of Wikipedia. In the second phase were executed SPARQL queries to DBpedia to gather information.

In the first phase, were executed these steps:

- List the districts of Portugal;
- Collect the id’s of the links from the pages of each district;
- Replace the id's for page titles;
- List all id's and titles for the pages with the same title page listed above (One id link could correspond to a not geographic type, but different entries could have the same title and one of this correspond to a geographic type);

- Insert in the previous list id's and titles of pages whose title is similar to those previously collected but with a termination "_(<term>)" (Some pages have the title <municipality>_(<district>) but in database that link appears with the title <municipality>);
- Filter links by type (“Munícipios_de_Portugal”);
- Remove the duplicate and incorrect entries (Some pages are the type “Munícipios_de_Portugal” but not are a geographical location, for example, it could be an entry for “clube_desportivo_<name>”);
- Inserts new entries that were not previously entered (for some reason the municipalities of Vila Real belongs to the type “Distrito_de_Vila_Real” unlike all the others municipalities that belong to type “Munícipios_de_Portugal”, it should be noted that, exception for these municipalities, are rare cases of municipalities that belong to the type “Distrito_de_ <distrito>”);
- List the parishes from each municipality;

Based on this structure is then possible to collect information about each geographical place. These data are collected in a cyclical manner, using queries-SPARQL. Two queries are made to DBpedia, the first is the redirect of the pages title from Portuguese to English, this process needs to be done because the structure was made from the Portuguese version of Wikipedia and so the pages title are in Portuguese and secondly the DBpedia do not have a Portuguese version. The other query can thus collect information from locations such as population, latitude and longitude.

GeoNames

The process of collecting information from Geonames is facilitated by the Web Services available. Two distinct Web Services are used, the first (Search) returns all entries in accordance with certain attributes (type and country). The other Web Service (hierarchy) returns the parent of an entry.

Similarity

With the collection of information from these two databases, is now permitted to pairing them, thus enabling a comparison of different geographic ontologies. Was done three different similarities measure, a comparison of the names of localities using the Levenshtein measure, a difference of populations and geographical distance.

Examples

DBpedia

SELECT * WHERE {

<http://dbpedia.org/resource/**<page_title>**> ?type ?value .

Filter(?type = dbpprop:populationTotal ||

?type = geo:lat ||

?type = geo:long) }

GeoNames

http://ws.geonames.org/search?country=PT &style=full&featureCode=**<type>**

http://ws.geonames.org/hierarchy?geonameId=**<geonameId>**

Similarity

delta_population = abs(log(pop1 / pop2))

distance = 6371 * acos(((cos(lon1 - lon2) * (cos(lat1 - lat2) + cos(lat1 + lat2))) + (cos(lat1 - lat2) - cos(lat1 + lat2))) / 2)

Discussion

The biggest critic I can do to my work is that ontologies similaries techniques were not applied, tecnhniques that weigh the similarity measures of a pair, with the similarities between the parents of this pair, but the process executed was done in order to allow these techniques in the future. Regarding the process executed to structure the data from DBpedia, to emphasize the fact that part of the dump of wikipedia is not consistent, eg two different pages (id's) can have the same title, this should mean that one URI could match different pages, what does not. Moreover, there are links hardly annotated, for

example in the Webpage of "Distrito_de_Vila_Real" there is a link to the municipality of "Mesão_Frio" and in the database the title of this page is "Mesão_Frio_ (Vila_Real).

One of the positive points of this work is to enable the improve of the structures of both databases, on one hand the GeoNames has some gaps in their structure between the administrative divisions and geographic locations, on the other hand DBpedia structure is virtually nonexistent and the executed work was curated by hand.

Speaking now on the results of similarity, I have more than 113 million results, being that, for each pair (more than 72M pairs) there is at least the result of the Levenshtein measure. An example of the results of similarity can be "Caldas da Rainha," because this location has all the information in both databases, containing the following results: for the method Levenshtein the similarity is equal to 1, the names are equal; in relation to geographical distance the obtained value was 0.8, this means that the distance between this location in the different databases is only 0.8 km; for differences in population the result was 0.6, indicating that the populations have the same magnitude but slightly different (26k for Geonames and 48k for DBpedia).

As mentioned previously, these results could allow a complement of data in both databases. One of the points where this can be well executed will be on the types of locations, the good structure created for DBpedia, can be extrapolated to the Geonames. In the example given above we can then indicate that “Caldas da Rainha” is a municipality, for this is necessary to make a match of the types in Geonames. Giving a concrete example, in the districts we have similarities values of 1 for the method Levenshtein, exception of two locations that I will refer later on, so you may have a correlation between the type, in DBpedia the types are "Distrito" while in Geonames are "first-order administrative division". From this correlation, and because we do not have all the information in both databases, we can still connect the “Distrito de Lisboa” (DBpedia) with the “Lisbon” (Geonames), Levenshtein similarity of 0.2 and the “Distrito de Bragança” (DBpedia) with "Bragança" (Geonames), Levenshtein similarity of 0.4.

Appendix

```
from MySQLdb import connect , IntegrityError
from math import acos , cos , log , radians
from xml.parsers.expat import ExpatError
from socket import setdefaulttimeout
from urllib import urlopen , quote
from xml.dom.minidom import parse
from configdb import *
from time import sleep
from os import system
from sys import argv

setdefaulttimeout( 0.5 * 60 )

def insertSQL( query , database = db ):
    conn = connect( host = "localhost" , user = user , passwd = password , db = database )
    meu_cursor = conn.cursor( )
    meu_cursor.execute( query )
    meu_cursor.close( )
    conn.commit( )
    conn.close( )

def querySQL( query , database = db ):
    conn = connect( host = "localhost" , user = user , passwd = password , db = database )
    meu_cursor = conn.cursor( )
    meu_cursor.execute( query )
    result = meu_cursor.fetchall( )
    meu_cursor.close( )
    conn.commit( )
    conn.close( )
    return result

def api( url ):
    try:
        f = urlopen( url )
        try: doc = parse( f )
        except ExpatError: doc = api( url )
        f.close( )
    except IOError: doc = api( url )
    return doc

def createDB( database = db ):
    conn = connect( host = "localhost" , user = user , passwd = password )
    meu_cursor = conn.cursor( )
    meu_cursor.execute( "drop database if exists " + database )
    meu_cursor.execute( "create database " + database )
    meu_cursor.close( )
    conn.commit( )
    conn.close( )

def createTables( ):
    conn = connect( host = "localhost" , user = user , passwd = password , db = db )
    meu_cursor = conn.cursor( )
    meu_cursor.execute( "drop table if exists Geoname" )
    meu_cursor.execute( "drop table if exists Anc_Geoname" )
    meu_cursor.execute( "drop table if exists GeoNetPT" )
    meu_cursor.execute( "drop table if exists Anc_GeoNetPT" )
    meu_cursor.execute( "drop table if exists DBpedia" )
    meu_cursor.execute( "drop table if exists Anc_DBpedia" )
    meu_cursor.execute( "drop table if exists Pair_gnm_pt" )
    meu_cursor.execute( "drop table if exists Similarity_gnm_pt" )
    meu_cursor.execute( "drop table if exists Pair_gnm_db" )
    meu_cursor.execute( "drop table if exists Similarity_gnm_db" )
    meu_cursor.execute( "drop table if exists Pair_pt_db" )
    meu_cursor.execute( "drop table if exists Similarity_pt_db" )
    meu_cursor.execute( "create table Geoname (id int primary key auto_increment, name varchar(200), lat float, lng float, pop int, type varchar(80), geonameId int unique, index(name(4)), index(lat), index(lng), index(type))" )
    meu_cursor.execute( "create table Anc_Geoname (id int primary key auto_increment, term1 int, term2 int, distance int, index(term1))" )
    meu_cursor.execute( "create table GeoNetPT (id int primary key auto_increment, name varchar(200), lat float, lng float, pop int, type varchar(80), geonetptId varchar(100) unique, index(name(4)), index(lat), index(lng), index(type))"
    meu_cursor.execute( "create table Anc_GeoNetPT (id int primary key auto_increment, term1 int, term2 int, distance int, index(term1))" )
    meu_cursor.execute( "create table DBpedia (id int primary key auto_increment, name varchar(200), lat float, lng float, pop int, type varchar(80), dbpediaId varchar(100), index(name(4)), index(lat), index(lng), index(type))" )
    meu_cursor.execute( "create table Anc_DBpedia (id int primary key auto_increment, term1 int, term2 int, distance int, index(term1))" )
    meu_cursor.execute( "create table Pair_gnm_pt (id int primary key auto_increment, term1 int, term2 int, index(term1), index(term2))" )
    meu_cursor.execute( "create table Similarity_gnm_pt (id int primary key auto_increment, pairId int, method varchar(32), similarity float, relation varchar(32), index(pairId), index(similarity), index(method))" )
    meu_cursor.execute( "create table Pair_gnm_db (id int primary key auto_increment, term1 int, term2 int, index(term1), index(term2))" )
    meu_cursor.execute( "create table Similarity_gnm_db (id int primary key auto_increment, pairId int, method varchar(32), similarity float, relation varchar(32), index(pairId), index(similarity), index(method))" )
    meu_cursor.execute( "create table Pair_pt_db (id int primary key auto_increment, term1 int, term2 int, index(term1), index(term2))" )
    meu_cursor.execute( "create table Similarity_pt_db (id int primary key auto_increment, pairId int, method varchar(32), similarity float, relation varchar(32), index(pairId), index(similarity), index(method))" )
    meu_cursor.close( )
    conn.commit( )
    conn.close( )

def levenshtein( s1 , s2 ):
    l1 = len( s1 )
    l2 = len( s2 )
    matrix = [ range( l1 + 1 ) ] * ( l2 + 1 )
    for z1 in range( l2 + 1 ):
      matrix[ z1 ] = range( z1 , z1 + l1 + 1 )
    for z1 in range( 0 , l2 ):
      for z2 in range( 0 , l1 ):
        if s1[ z2 ] == s2[ z1 ]:
          matrix[ z1 + 1 ][ z2 + 1 ] = min( matrix[ z1 + 1 ][ z2 ] + 1 , matrix[ z1 ][ z2 + 1 ] + 1 , matrix[ z1 ][ z2 ] )
        else:
          matrix[ z1 + 1 ][ z2 + 1 ] = min( matrix[ z1 + 1 ][ z2 ] + 1 , matrix[ z1 ][ z2 + 1 ] + 1 , matrix[ z1 ][ z2 ] + 1 )
    return 1 - float( matrix[ l2 ][ l1 ] ) / max( l1 , l2 )

def delta_geo( geo1 , geo2 ):
  """Return the difference between the two latitudes or longitude.
  """
  return abs( geo1 - geo2 )

earth_radius = 6371
def distance( lat1 , lon1 , lat2 , lon2 ):
  """Return the minimum distance in Earth's spherical surface between the points
  defined by coordinates (lat1, lon1) and (lat2, lon2). The values must be given
  in degrees.
  """
  # Convert degrees to radians
  # - x: latitude
  # - y: longitude
  x1 , y1 , x2 , y2 = ( radians( i ) for i in ( lat1 , lon1 , lat2 , lon2 ) )

  p1 = cos( y1 - y2 )
  p2 = cos( x1 - x2 )
  p3 = cos( x1 + x2 )

  inner_product = ( ( p1 * ( p2 + p3 ) ) + ( p2 - p3 ) ) / 2
  return earth_radius * acos( inner_product )
```

```
def d_population( pop1 , pop2 ):
  """Return the difference between the two population values given. The result
  is the logarithm of the division of the two values, which means that small
  difference become negligible and differences of order of magnitude are given
  the most consideration.
  """
  return abs( log( float( pop1 ) / pop2 ) )

def deltas( db1 , db2 ):
    print "\tprogress" ,
    if db1 == "gnm": database1 = "Geoname"
    elif db1 == "pt": database1 = "GeoNetPT"
    else:
        print "Incorrect Database."
        return ""
    if db2 == "pt": database2 = "GeoNetPT"
    elif db2 == "db": database2 = "DBpedia"
    else:
        print "Incorrect Database."
        return ""
    query = "select count(*) from Pair_" + db1 + "_" + db2
    num_pairs = querySQL( query )[ 0 ][ 0 ]
    for i in xrange( 0 , num_pairs , 10000 ):
        print "%f%%"%( float( i ) / num_pairs ) ,
        query = "select p.id, " + db1 + ".name, " + db2 + ".name, " + db1 + ".lat, " + db1 + ".lng, " + db2 + ".lat, " + db2 + ".lng, " + db1 + ".pop, " + db2 + ".pop from (select * from Pair_" + db1 + "_" + db2 + " limit %i, 10000) p, '
        pairs = querySQL( query )
        for pair in pairs:
            query = "insert into Similarity_" + db1 + "_" + db2 + " (pairId, method, similarity) values (%i,'levenshtein',%i)"%( pair[ 0 ] , levenshtein( pair[ 1 ] , pair[ 2 ] ) )
            insertSQL( query )
            if pair[ 3 ] and pair[ 5 ]:
                query = "insert into Similarity_" + db1 + "_" + db2 + " (pairId, method, similarity) values (%i,'delta_lat',%i)"%( pair[ 0 ] , delta_geo( pair[ 3 ] , pair[ 5 ] ) )
                insertSQL( query )
            if pair[ 4 ] and pair[ 6 ]:
                query = "insert into Similarity_" + db1 + "_" + db2 + " (pairId, method, similarity) values (%i,'delta_lng',%i)"%( pair[ 0 ] , delta_geo( pair[ 4 ] , pair[ 6 ] ) )
                insertSQL( query )
            if pair[ 3 ] and pair[ 5 ] and pair[ 4 ] and pair[ 6 ]:
                query = "insert into Similarity_" + db1 + "_" + db2 + " (pairId, method, similarity) values (%i,'geo_dist',%i)"%( pair[ 0 ] , distance( pair[ 3 ] , pair[ 4 ] , pair[ 5 ] , pair[ 6 ] ) )
                insertSQL( query )
            if pair[ 7 ] and pair[ 8 ]:
                query = "insert into Similarity_" + db1 + "_" + db2 + " (pairId, method, similarity) values (%i,'delta_pop',%i)"%( pair[ 0 ] , delta_geo( pair[ 7 ] , pair[ 8 ] ) )
    print "complete."

def pairDB( ):
    insertSQL( "insert into Pair_gnm_pt (term1, term2) select gnm.id, pt.id from Geoname gnm, GeoNetPT pt" )
    insertSQL( "insert into Pair_gnm_db (term1, term2) select gnm.id, db.id from Geoname gnm, DBpedia db" )
    insertSQL( "insert into Pair_pt_db (term1, term2) select pt.id, db.id from GeoNetPT pt, DBpedia db" )

def dbpediaCrawler( ):
    createDB( "ptwiki" )
    system( "mysql -u " + user + " -p" + password +  " ptwiki < /disk1/ptwiki/ptwiki-latest-page.sql" )
    system( "mysql -u " + user + " -p" + password +  " ptwiki < /disk1/ptwiki/ptwiki-latest-categorylinks.sql" )
    system( "mysql -u " + user + " -p" + password +  " ptwiki < /disk1/ptwiki/ptwiki-latest-pagelinks.sql" )
    queries = [ 'create table temp1 as select c.cl_from distrito_id, c.cl_sortkey distrito, p.page_title distrito_title from categorylinks c, page p where c.cl_from = p.page_id and c.cl_to = "Distritos_de_Portugal" and c.cl_sortkey != "-
                'create table temp2 as select t.*, l.pl_from concelho_id from temp1 t, pagelinks l where t.distrito_title = l.pl_title' ,
                'create table temp3 as select t.distrito_id, t.distrito, t.distrito_title, p.page_title concelho_title from temp2 t, page p where t.concelho_id = p.page_id' ,
                'create table temp4 as select distinct t.distrito_id, t.distrito, t.distrito_title, p.page_id concelho_id, t.concelho_title from temp3 t, page p where t.concelho_title = p.page_title' ,
                'alter table temp4 add unique(distrito_id,concelho_id)' ,
                'insert ignore into temp4 select distinct t.distrito_id, t.distrito, t.distrito_title, p.page_id concelho_id, t.concelho_title from temp3 t, page p where SUBSTRING( t.concelho_title, 1, LOCATE( " (" , t.concelho_title ) -
                'create table temp5 as select t.distrito_id, t.distrito, t.distrito_title, t.concelho_id, c.cl_sortkey concelho, t.concelho_title from temp4 t, categorylinks c where t.concelho_id = c.cl_from and c.cl_to like "%Unio%ptica
                'delete from temp5 where (distrito_id = 1342509 and concelho_id = 384125) or (distrito_id = 1342773 and (concelho_id = 43811 or concelho_id = 283853)) or (distrito_id = 1343010 and (concelho_id = 331954 or concelho_id = "
                'alter table temp5 add unique(distrito_id,concelho_id)' ,
                'insert ignore into temp5 select distinct distrito_id, distrito, distrito_title, c1_from concelho_id, c1_sortkey concelho, page_title concelho_title from categorylinks, page, temp5 where (distrito_id = 1342797 and page_i
                'update temp5, page set concelho_title = page_title where distrito_id = 1343040 and concelho_id = "1297537 and page_id = 12411" ;
                'create table temp6 as select t.*, c.cl_from freguesia_id from temp5 t, categorylinks c where convert(c.cl_sortkey using utf8) = convert(if(LOCATE( " (" , t.concelho ),SUBSTRING( t.concelho , 1, LOCATE( " (" , t.concelho
                'alter table temp6 add unique(distrito_id,concelho_id)' ,
                'insert ignore into temp6 select t.*, page_id from temp5 t, page where (t.concelho_id = 1297779 and page_id = 1297778) or (t.concelho_id = 655065 and page_id = 1593665) or (t.concelho_id = 332165 and page_id = 1594620) o
                'create table temp7 as select t.*, p.page_title freguesia from temp6 t, page p where p.page_id = t.freguesia_id' ,
                'create table temp8 as select t.distrito_id, t.distrito, t.distrito_title, t.concelho_id, t.concelho, t.concelho_title, c.cl_from freguesia_id, c.cl_sortkey freguesia from temp7 t, categorylinks c where t.freguesia = c.c
                'create table temp9 as select t.*, p.page_title freguesia_title from temp8 t, page p where p.page_id = t.freguesia_id' ,
                'delete from temp9 where freguesia_id = 2549310 or freguesia_id = 19301 or freguesia_id = 10057 or freguesia_id = 219834 or freguesia_id = 1293909 or freguesia_id = 1941715 or freguesia_id = 2012299 or freguesia_id = 2383
                'insert into temp9 select distinct distrito_id, distrito, distrito_title, concelho_id, concelho, concelho_title, page_id, c1_sortkey, page_title from temp8, page, categorylinks where page_id = c1_from and ((concelho_id =
                'delete from temp9 where (concelho_id = 313529 or concelho_id = 474603 or concelho_id = 1467981) and freguesia != " "' ,
                'insert into temp9 select distinct distrito_id, distrito, distrito_title, concelho_id, concelho, concelho_title, page_id, c1_sortkey, page_title from temp8, page, categorylinks where page_id = c1_from and concelho_id = 1
    for query in queries:
        insertSQL( query , "ptwiki" )
    print "\tinserting data from Portugal"
    url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0APREFIX+rd
    doc = api( url )
    doc = doc.getElementsByTagName( "binding" )
    check_lat = False
    check_lng = False
    insert = 'insert into DBpedia (name, type, dbpediaId,'
    values = ') values ( "Portugal", "Pais", "Portugal",'
    for i in range( 0 , len( doc ) , 2 ):
        result_type = doc[ i ].getElementsByTagName( "uri" )[ 0 ].childNodes[ 0 ].data
        result_value = doc[ i + 1 ].childNodes[ 0 ].childNodes[ 0 ].data
        if result_type == "http://dbpedia.org/property/populationCensus":
            insert += "pop,"
            values += result_value + ","
        elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#lat" and not check_lat:
            insert += "lat,"
            values += result_value + ","
            check_lat = True
        elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#long" and not check_lng:
            insert += "lng,"
            values += result_value + ","
            check_lng = True
    query = insert[ : -1 ] + values[ : -1 ] + ")"
    insertSQL( query )
    query = "insert into Anc_DBpedia (term1,term2,distance) values (1 ,1, 0)"
    insertSQL( query )
    query = "select distinct distrito_id, distrito, distrito_title from temp9"
    distritos = querySQL( query , "ptwiki" )
    for distrito in distritos:
        print "\tinserting data from" , quote( quote( distrito[ 2 ] ) )
        url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0APREFI
        doc = api( url )
        doc = doc.getElementsByTagName( "uri" )
        insert = 'insert into DBpedia (name,type,'
        values = ') values (convert(" + unicode( distrito[ 2 ] , "utf-8" ).replace( "_" , " " ) + '" using latin1),"Distrito",'
        try:
            url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0AP
            insert += 'dbpediaId,'
            values += '"' + doc[ 1 ].childNodes[ 0 ].data.replace( "http://dbpedia.org/resource/" , "" ) + '",'
        except: IndexError:
            url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0AP
```

```
            insert += 'dbpediaId,'
            values += '"' + quote( quote( distrito[ 2 ] ) ) + '",'
        check_lat = False
        check_lng = False
        doc = api( url )
        doc = doc.getElementsByTagName( "binding" )
        for i in range( 0 , len( doc ) , 2 ):
            result_type = doc[ i ].childNodes[ 0 ].childNodes[ 0 ].data
            result_value = doc[ i + 1 ].childNodes[ 0 ].childNodes[ 0 ].data
            if result_type == "http://dbpedia.org/property/populationTotal":
                insert += "pop,"
                values += str( float( result_value ) * 1000 ) + ","
            elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#lat" and not check_lat:
                insert += "lat,"
                values += result_value + ","
                check_lat = True
            elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#long" and not check_lng:
                insert += "lng,"
                values += result_value + ","
                check_lng = True
        query = insert[ : - 1 ] + values[ : - 1 ] + ")"
        insertSQL( query )
        query = "select max(id) from DBpedia"
        distrito_id = str( querySQL( query )[ 0 ][ 0 ] )
        query = "insert into Anc_DBpedia (term1,term2,distance) select " + distrito_id + ", a.term2, a.distance + 1 from DBpedia d, Anc_DBpedia a where d.id=1 and d.id = a.term1 union select " + distrito_id + "," + distrito_id + ",0"
        insertSQL( query )
        query = "select distinct concelho_id, concelho, concelho_title from temp9 where distrito_id=" + str( distrito[ 0 ] )
        concelhos = querySQL( query , "ptwiki" )
        for concelho in concelhos:
            print "\t\tinserting data from" , quote( quote( concelho[ 2 ] ) )
            url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D%0API
            doc = api( url )
            doc = doc.getElementsByTagName( "uri" )
            insert = 'insert into DBpedia (name,type,'
            values = ') values (convert(' + unicode( concelho[ 2 ] , "utf-8" ).replace( "_" , " " ) + ' using latin1),"Concelho",'
            try:
                url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D
                insert += 'dbpediaId,'
                values += '"' + doc[ 1 ].childNodes[ 0 ].data.replace( "http://dbpedia.org/resource/" , "" ) + '",'
            except IndexError:
                url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D
                insert += 'dbpediaId,'
                values += '"' + quote( quote( concelho[ 2 ] ) ) + '",'
            check_lat = False
            check_lng = False
            doc = api( url )
            doc = doc.getElementsByTagName( "binding" )
            for i in range( 0 , len( doc ) , 2 ):
                result_type = doc[ i ].childNodes[ 0 ].childNodes[ 0 ].data
                result_value = doc[ i + 1 ].childNodes[ 0 ].childNodes[ 0 ].data
                if result_type == "http://dbpedia.org/property/populationTotal":
                    insert += "pop,"
                    values += str( float( result_value ) * 1000 ) + ","
                elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#lat" and not check_lat:
                    insert += "lat,"
                    values += result_value + ","
                    check_lat = True
                elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#long" and not check_lng:
                    insert += "lng,"
                    values += result_value + ","
                    check_lng = True
            query = insert[ : - 1 ] + values[ : - 1 ] + ")"
            insertSQL( query )
            query = "select max(id) from DBpedia"
            concelho_id = str( querySQL( query )[ 0 ][ 0 ] )
            query = "insert into Anc_DBpedia (term1,term2,distance) select " + concelho_id + ", a.term2, a.distance + 1 from DBpedia d, Anc_DBpedia a where d.id=" + distrito_id + " and d.id = a.term1 union select " + concelho_id + "," +
            insertSQL( query )
            query = "select distinct freguesia_id, freguesia, freguesia_title from temp9 where concelho_id=" + str( concelho[ 0 ] )
            freguesias = querySQL( query , "ptwiki" )
            for freguesia in freguesias:
                print "\t\t\tinserting data from" , quote( quote( freguesia[ 2 ] ) )
                url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3E%0D
                doc = api( url )
                doc = doc.getElementsByTagName( "uri" )
                insert = 'insert into DBpedia (name,type,'
                values = ') values (convert(' + unicode( freguesia[ 2 ] , "utf-8" ).replace( "_" , " " ) + ' using latin1),"Freguesia",'
                try:
                    url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3
                    insert += 'dbpediaId,'
                    values += '"' + doc[ 1 ].childNodes[ 0 ].data.replace( "http://dbpedia.org/resource/" , "" ) + '",'
                except IndexError:
                    url = "http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=PREFIX+owl%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2002%2F07%2Fowl%23%3E%0D%0APREFIX+xsd%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema%23%3
                    insert += 'dbpediaId,'
                    values += '"' + quote( quote( freguesia[ 2 ] ) ) + '",'
                check_lat = False
                check_lng = False
                doc = api( url )
                doc = doc.getElementsByTagName( "binding" )
                for i in range( 0 , len( doc ) , 2 ):
                    result_type = doc[ i ].childNodes[ 0 ].childNodes[ 0 ].data
                    result_value = doc[ i + 1 ].childNodes[ 0 ].childNodes[ 0 ].data
                    if result_type == "http://dbpedia.org/property/populationTotal":
                        insert += "pop,"
                        values += str( float( result_value ) * 1000 ) + ","
                    elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#lat" and not check_lat:
                        insert += "lat,"
                        values += result_value + ","
                        check_lat = True
                    elif result_type == "http://www.w3.org/2003/01/geo/wgs84_pos#long" and not check_lng:
                        insert += "lng,"
                        values += result_value + ","
                        check_lng = True
                query = insert[ : - 1 ] + values[ : - 1 ] + ")"
                insertSQL( query )
                query = "select max(id) from DBpedia"
                freguesia_id = str( querySQL( query )[ 0 ][ 0 ] )
                query = "insert into Anc_DBpedia (term1,term2,distance) select " + freguesia_id + ", a.term2, a.distance + 1 from DBpedia d, Anc_DBpedia a where d.id=" + concelho_id + " and d.id = a.term1 union select " + freguesia_id +
                insertSQL( query )

def geonetptCrawler( typeOf = "pais-AIPAI" , local = "portugal-AF418745" , anc = 0 ):
    print "\tinserting" , typeOf , "for" , local , "data."
    check = True
    name_pop = "http://aldb.di.fc.ul.pt/dbpedia/sparql?default-graph-uri=http%3A%2F%2Fgeonetpt02.aldb.di.fc.ul.pt&query=select%0D%0Atype%0D%0Avalue+where+{%0D%0Aentity+gn%0D%0Atype+<http%0D%0A%2F%2Faldb.di.fc.ul.pt%2Faldb%2Fpublications%2
    doc_np = api( name_pop )
    doc_np = doc_np.getElementsByTagName( "literal" )
    lat_lng = "http://aldb.di.fc.ul.pt/dbpedia/sparql?default-graph-uri=http%3A%2F%2Fgeonetpt02.aldb.di.fc.ul.pt&query=select%0D%0Alat%0D%0Alng+where+{%0D%0Aentity+gn%0D%0Atype+<http%0D%0A%2F%2Faldb.di.fc.ul.pt%2Faldb%2Fpublications%2F200
```

```
    doc_ll = api( lat_lng )
    doc_ll = doc_ll.getElementsByTagName( "literal" )
    infoType = "http://wldb.di.fc.ul.pt/dbpedia/sparql?default-graph-uri=http%3A%2F%2Fgeonetpt02.wldb.di.fc.ul.pt&query=select+%3Fname+where+{%3Fentity+gn%3Atype+<http%3A%2F%2Fwldb.di.fc.ul.pt%2Fwldb%2Fpublications%2F2009%2F
    doc_t = api( infoType )
    doc_t = doc_t.getElementsByTagName( "literal" )
    try:
        insert = 'insert into GeoNetPT (name,'
        values = '( values ("%s",'%doc_np[ 0 ].childNodes[ 0 ].data.replace('"','\\"')
        try:
            values += '%s, %s,'%( doc_ll[ 0 ].childNodes[ 0 ].data , doc_ll[ 1 ].childNodes[ 0 ].data )
            insert += 'lat, lng,'
        except IndexError: ""
        try:
            values += '%s,'%doc_np[ 1 ].childNodes[ 0 ].data
            insert += 'pop,'
        except IndexError: ""
        try:
            values += '"%s",'%doc_t[ 0 ].childNodes[ 0 ].data
            insert += 'type,'
        except IndexError: ""
        insert += 'geonetptId'
        values += '"%s")'%( "<http%3A%2F%2Fwldb.di.fc.ul.pt%2Fwldb%2Fpublications%2F2009%2F10%2Fgeo-net-pt-02%23%s>"%local )
        query = insert + values
    except IndexError: check = False
    if check:
        try: insertSQL( query )
        except IntegrityError: check = False
        if check:
            pai = querySQL( "select max( id ) from GeoNetPT" )[ 0 ][ 0 ]
            insertSQL( "insert into Anc_GeoNetPT (term1, term2, distance) select %i, a.term2, a.distance + 1 from Anc_GeoNetPT a, GeoNetPT pt where pt.id=%i and pt.id=a.term1;"%( pai , anc ) )
            insertSQL( "insert into Anc_GeoNetPT (term1, term2, distance) values ( %i, %i, 0)"%( pai , pai ) )
            url_anc = "http://wldb.di.fc.ul.pt/dbpedia/sparql?default-graph-uri=http%3A%2F%2Fgeonetpt02.wldb.di.fc.ul.pt&query=select+%3Fvalue%3Fvalue_type+where+{%3Fentity+gn%3Atype+<http%3A%2F%2Fwldb.di.fc.ul.pt%2Fwldb%2Fp
            i = 0
            while True:
                doc_anc = api( url_anc%( typeOf , local , i ) )
                doc_anc = doc_anc.getElementsByTagName( "result" )
                for anc in doc_anc:
                    anc_uri = anc.getElementsByTagName( "uri" )
                    typeOf_anc = anc_uri[ 1 ].childNodes[ 0 ].data
                    typeOf_anc = typeOf_anc[ typeOf_anc.find( "#" ) + 1 : ]
                    local_anc = anc_uri[ 0 ].childNodes[ 0 ].data
                    local_anc = local_anc[ local_anc.find( "#" ) + 1 : ]
                    if local_anc.find( "'" ) == -1:
                        geonetptCrawler( typeOf_anc , local_anc , pai )
                if len( doc_anc ) < 100:
                    break
                else:
                    i += 100

def geonameCrawler( ):
    admin = [ "admin1Code=01" , "admin1Code=03" , "admin1Code=04" , "admin1Code=05" , "admin1Code=06" , "admin1Code=07" , "admin1Code=08" , "admin1Code=09" , "admin1Code=10" , "admin1Code=11" , "admin1Code=13" , "admin1Code=
    for code in [ "PCLI" , "ADM1" , "ADM2" , "ADM3" , "ADM4" , "PPLC" , "PPLA" , "PPLA2" , "PPLA3" , "PPL" , "PPLQ" , "PPLL" , "PPLR" , "PPLX" ]:
        print "\tinserting" , code , "data." ,
        i = 0
        j = 0
        while True:
            print i ,
            if i != 0:
                print "(waiting)" ,
                sleep( 72 )
            uri = "http://ws.geonames.org/search?country=PT&featureCode=%s&style=full&maxRows=100&startRow=%i"%( code , i )
            if code == "PPL":
                uri += admin[ j ]
            doc1 = api( uri )
            geonames = doc1.getElementsByTagName( "geoname" )
            for geoname in geonames:
                if geoname.getElementsByTagName( "countryCode" )[ 0 ].childNodes[ 0 ].data == "PT":
                    try: query = 'insert into Geoname ( name , lat , lng , pop , geonameId , type ) values ( "%s" , %s , %s , %s , %s , "%s" )'%( geoname.getElementsByTagName( "name" )[ 0 ].childNodes[ 0 ].data , geoname.getElementsByTa
                    except: query = 'insert into Geoname ( name , lat , lng , geonameId , type ) values ( "%s" , %s , %s , %s , "%s" )'%( geoname.getElementsByTagName( "name" )[ 0 ].childNodes[ 0 ].data , geoname.getElementsByTagName( "
                    insertSQL( query )
                    doc2 = api( "http://ws.geonames.org/hierarchy?geonameId=%s"%geoname.getElementsByTagName( "geonameId" )[ 0 ].childNodes[ 0 ].data )
                    doc2 = doc2.getElementsByTagName( "geoname" )
                    for anc in range( 2 , len( doc2 ) ):
                        query = 'insert into Anc_Geoname ( term1 , term2 , distance ) select max( g2.id ) , g1.id , %i from Geoname g1 , Geoname g2 where g1.geonameId = %s group by g1.id'%( len( doc2 ) - 1 - anc , doc2[ anc ].getElement
                        insertSQL( query )
            if len( geonames ) < 100:
                if code == "PPL" and j < len( admin ) - 1:
                    i = 0
                    j += 1
                else:
                    break
            else:
                i += 100
        print "finish!"
        sleep( 72 )

for i in range( 1 , len( argv ) ):
    if argv[ i ] == "db":
        print "creating DB."
        createDB( )
    elif argv[ i ] == "tbl":
        print "creating Tables."
        createTables( )
    elif argv[ i ] == "gnm":
        print "inserting Geoname data."
        geonameCrawler( )
    elif argv[ i ] == "gnpt":
        print "inserting GeoNetPT data."
        geonetptCrawler( )
    elif argv[ i ] == "db":
        print "inserting DBpedia data."
        dbpediaCrawler( )
    elif argv[ i ] == "pair":
        print "pairing databases."
        pairDB( )
    elif argv[ i ] == "sim_gnm_pt":
        print "calculating simularity between Geoname and GeoNetPT."
        deltas( "gnm" , "pt" )
    elif argv[ i ] == "sim_gnm_db":
        print "calculating simularity between Geoname and DBpedia."
        deltas( "gnm" , "db" )
    elif argv[ i ] == "sim_pt_db":
        print "calculating simularity between GeoNetPT and DBpedia."
        deltas( "pt" , "db" )

if len( argv ) == 1:
    print "creating DB."
    createDB( )
    print "creating Tables."
    createTables( )
    print "inserting Geoname data."
    geonameCrawler( )
    print "inserting GeoNetPT data."
    geonetptCrawler( )
    print "inserting DBpedia data."
    dbpediaCrawler( )
    print "pairing databases."
    pairDB( )
    print "calculating simularity between Geoname and GeoNetPT."
    deltas( "gnm" , "pt" )
    print "calculating simularity between Geoname and DBpedia."
    deltas( "gnm" , "db" )
    print "calculating simularity between GeoNetPT and DBpedia."
    deltas( "pt" , "db" )
```